REMARKS/ARGUMENTS

In the Office Action, the Examiner made a request for information and maintained the rejection of the claims.

Request for Information under 37 CFR 1.105

In response to the Examiner's Requirement for Information under 37 CFR 1.105, the Applicant hereby submits herewith copies of the "Old Jbug schedule," and "Jbug FCS schedule" documents (Office Action, paragraph 2a on page 2).

In addition, the Applicant submits that the claimed feature of "automatically generating front-end and back-end debugger program portions based on parsing of a formal specification" was tested during "Beta 1" and "Beta 2" experimental testing releases (Office Action, paragraph 2b on page 2).

The Examiner has also requested that the Applicant provide copies, and "specify the relevant section(s), of any available documentation supporting this feature in identified software products(s)" (Office Action, paragraph 2b on page 3). The Applicant respectfully submits that it is unclear what information the Examiner is attempting to request. Nevertheless, the Applicant hereby submits herewith another declaration under 37 CFR § 1.132 (Declaration B), signed by Mr. Robert Field, an inventor of the claimed invention set forth in the above referenced application.

The Examiner has also requested the "details of the supervision and control polices governing the Beta 1 testing and Beta 2 testing." The Applicant has already submitted a copy of the Jbug Beta Software License which was granted for a sixty (60) day test period. This document details the limitations and restrictions placed on testing of the experimental releases of the Java Platform Debugger Architecture (Jbug) (see, for example, section 2(c) of the Jbug Beta Software License). The Applicant further submits Declaration B which attests that Beta 1 testing and Beta 2 testing were conducted under direct supervision and control of Mr. Robert Field (paragraph 14).

Rejection of claims 20 and 24-32

In the Office Action, the Examiner has rejected claims 20 and 24-32 under 35 U.S.C. 102(b) based on public use of the invention, or alternatively under 35 U.S.C.

103(a) as obvious over allegation of public use in view of *Aho et al.* The Applicant reiterates the arguments previously submitted and requests that the Examiner's rejection be withdrawn. In addition, the Applicant hereby submits another declaration (Declaration C) signed by Mr. Robert Field which attests that the concept of a formal specification was conceived after the presentation which is the basis of the Examiner's rejection, and further evidences that the claimed invention was NOT obvious over alleged public use in view of general parsing techniques of *Aho et al.* The Applicant has already addressed the Examiner's rejection under 35 U.S.C. 103. A portion of the Applicant's arguments addressing the Examiner's rejection of claims under 35 U.S.C. 103 (a) over allegation of public use in view of *Aho et al.* are reproduced below.

The Examiner has not met his burden to establish a prima facie case of obviousness because the Examiner has NOT even pointed to a motivation or suggestion for automatically generating a single debugging component based on a formal specification. Instead, the Examiner has merely asserted that the alternative to compiling is writing code directly in assembly language or binary which is typically impractical.

Contrary to the Examiner's assertion, it is respectfully submitted that there are other practical alternatives aside from writing the code directly in assembly language or binary to generate the front-end and/or back-end debugging components in the context of the claimed invention. For example, each of the front-end and back-end components may be manually generated using a different specification. In other words, it is not required as the only alternative to writing code directly in assembly language or binary that the same formal specification be used to automatically generate both the front-end and back-end components. In fact, prior to the invention, the front-end and back-end of the multi-platform debugger were not both automatically generated based on the same formal specification. As noted in the specification, the code for the back-end processing module was manually written (Specification, paragraph 33). But this doesn't mean that this code had to be written in assembly or binary code. As noted in the specification, a "include" file for the back-end was manually written in a high-level programming language (e.g., the C programming language) (Specification, paragraph 33).

Thus, contrary to the Examiner's assertion, not having to code in binary or assembly language is NOT sufficient motivation or suggestion for automatically generating both back-end and front-end debugging components from the same formal

specification. One motivation or suggestion for this automation is noted in the specification, namely, a desire to address compatibility and documentation issue in an effort to promote the evolution of multi-platform debugging systems (Specification, paragraph 33). To achieve this, a formal specification was conceived which, among other things, can be used to automatically generate both the front-end and back-end debugging components in the context of the claimed invention.

Accordingly, it is submitted that the Examiner has misconstrued what may be a practical alternative to the automation in the context of the claimed invention and, as a result, the Examiner has clearly failed to provide a motivation or suggestion for the automation features of the claimed invention.

Still further, it is respectfully submitted that in order to establish a *prima facie* case of obviousness, all claim limitations must be taught or suggested by the cited art (MPEP § 21403.03). However, the combination of the "slideshow" and conventional compiling techniques do not teach or suggest these claimed limitations. The Applicant has submitted that the "slideshow" does not teach many of the recited features of the claimed invention (see, for example, Appeal Brief). As will be discussed, the conventional compiling techniques of *Aho et al.* also fail to teach or suggest these features. In fact, these conventional compiling techniques do not even address the specific problems encountered in multi-platform debugging environments.

It should be noted that other conventional techniques also fail to address these specific problems. As noted in the specification, languages exist for the specification of inter-process/object communication, such as the Interface Definition Language (IDL) which is part of the Common Object Request Broker Architecture (CORBA), developed by the Object Management Group (OMG). These languages are compiled (i.e., by an IDL compiler) to produce stubs for the client side of communication and skeletons for the server side. However, such languages also fail to address the problems associated with generating protocol compliant debugger code (Specification, paragraph 5).

The conventional compiling techniques of *Aho et al.* fail to even address specific problems that are encountered in multi-platform debugging environments and as such cannot possibly teach or suggest many of the claimed features in the context of the invention. First, it should also be noted that multi-platform debugging environments introduce several problems that cannot merely be addressed by general conventional

compilation techniques which the Examiner has relied on. Typically, a front-end component (debugger) is written in one programming language (e.g., Java™ programming language) while the back-end (debuggee) is written in another (e.g., C programming language). Yet both of these components are to be compliant with the same formal specification and are to be automatically generated based on the same formal specification in accordance with the claimed invention (see, for example, claim 20).

In addition, it is respectfully submitted that the conventional compiling techniques of *Aho et al.* do NOT even address these claimed features because, among other things, a conventional compiler of *Aho et al.* produces code only for a single programming language that is, in turn, executed on a particular platform (hardware and/or operating system). In other words, conventional compiling techniques do not teach or suggest generating different programming codes for different components based on the same input.

Also, conventional compiling techniques of *Aho et al.* relied on by the Examiner do NOT teach or suggest generating debugging components that communicate with each other in accordance with a specific communication protocol. In fact, conventional compiling techniques of *Aho et al.* generally teach generating binary or assembly code for a high-level programming language. These techniques, however, do not teach producing program code for a platform independent component (e.g., front-end debugger code written in Java™ programming language), or a definition file for a platform dependent programming language (e.g., "Constant.h" file for a back-end written in C programming language), or a human readable specification in a markup language (e.g., HTML documents). Moreover, the conventional compiling techniques of *Aho et al.* do not teach or suggest automatically generating all of these components from a single specification.

Clearly, conventional compiling techniques of *Aho et al.* merely teach general parsing and compiling of a program written in a programming language to generate binary or assembly code. Conventional compiling techniques, however, do not teach or suggest: automatically generating, based on the same specification, various debugger components that communicate with each other. Hence, even assuming purely for the sake of argument that a communication protocol for a multi-platform debugging environment was known, one of ordinary skilled in the art at the time of invention could

NOT merely adapt the introductory compiling principles of *Aho et al.* to automatically generate a front-end debugger that provides a high-level debugging interface in a platform independent language and a back-end debugger program associated with a specific platform from the same formal specification, such that these components are compatible with each other and comply with the same formal specification, and yet function and meet the requirements of a debugging environment using a high level debugging communication protocol provided for communication between first and second virtual machines that respectively support the front-end and back-end debugger components (claim 20).

To achieve the claimed features, a purely declarative language with syntax, grammar and semantics suitable for multi-platform debugging systems can be used in accordance with one embodiment of the invention. The suitable syntax, grammar and semantics of a Java™ Debugger Wire protocol is illustrated on pages 13-15 of the specification in accordance with one embodiment of the invention. Clearly, introductory compiling techniques of *Aho et al.* do NOT teach or suggest suitable syntax, grammar and semantics for a debugging wire protocol or any other reasonable technique for automatically generating a front-end debugger program portion and a back-end debugger program that are generated from the same formal specification in the context of the claimed invention.

Accordingly, it is submitted that the rejection of claims under U.S.C. § 103 (a) is clearly improper.

In view of the foregoing arguments, it is earnestly believed that is evident that the "slideshow" and *Aho et al.,* taken alone or in any proper combination, do NOT teach or suggest many other claimed features recited in dependent claims 24-32. Therefore, without further discussion, it is respectfully submitted that the "slideshow" and *Aho et al.,* taken alone or in any proper combination, do not teach or suggest many other recited features of dependent claims 24-32.

These features, for example, include the following: a front-end processing module that operates to send events that are generated in a virtual machine to a front-end debugger program portion via a back-end debugger program code portion (claim 24); a front-end processing module that performs one or more of the following operations: read and parse events from the back-end debugger code portion, convert

the events from a first format into a second format which is compatible with a front-end debugger code portion, and queue the events (claim 25); a front-end processing module that performs operations related to requests made through a front-end debugger program by a debugger application program (claim 26); a front-end processing module that performs one or more of the following operations: write formatted requests, send the formatted requests to the back-end debugger code portion, associate at least one reply with the formatted requests, and read and parse the at least one reply, and deliver the at least one reply to an appropriate requester (claim 27); and a front-end debugger program portion that includes a class which is used by the front-end debugger program portion to send and receive information over the debugging communication protocol (claim 32).

These features also include: a back-end processing module that performs operations related to event processing and request processing (claim 28), or performs event processing operations including: sending an event which was generated through a virtual machine debugging interface to a front-end debugging portion (claim 29), or such that the request processing operations include one or more of the following operations: reading and parsing formatted requests from a front-end debugger program portion, forwarding the requests to the back-end debugger program code portion, and sending the reply to the requests to the front-end debugger program portion (claim 30).

Conclusion

Accordingly, it is requested that the Examiner withdraw the pending rejections under 35 U.S.C. §102(b) or 103 (a). Applicants hereby petition for any extension of time which may be required to maintain the pendency of this case, and any required fees for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P252). Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Ramin Mahboubian
Reg. No. 44,890

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300